Contention Resolution Without Collision Detection: Constant Throughput *And* Logarithmic Energy

Gianluca De Marco
Department of Computer Science, University of Salerno, Italy

Dariusz R. Kowalski
School of Computer and Cyber Sciences, Augusta University, GA, USA

Grzegorz Stachowiak
Institute of Computer Science, University of Wrocław, Poland

Abstract

A shared channel, also called a multiple access channel, is among the most popular and widely studied models of communication in distributed computing. An unknown number of stations (potentially unbounded) is connected to the channel and can communicate by transmitting and listening. A message is successfully transmitted on the channel if and only if there is a unique transmitter at that time; otherwise the message collides with some other transmission and nothing is sensed by the participating stations. We consider the general framework without collision detection and in which any participating station can join the channel at any moment. The contention resolution task is to let each of the contending stations to broadcast successfully its message on the channel.

In this setting we present the first algorithm which exhibits asymptotically optimal $\Theta(1)$ throughput and only an $O(\log k)$ energy cost, understood as the maximum number of transmissions performed by a single station (where k is the number of participating stations, initially unknown). We also show that such efficiency cannot be reproduced by non-adaptive algorithms, *i.e.*, whose behavior does not depend on the channel history (for example, classic backoff protocols). Namely, we show that non-adaptive algorithms cannot simultaneously achieve throughput $\Omega\left(\frac{1}{\text{polylog}(k)}\right)$ and energy $O\left(\frac{\log^2 k}{(\log\log k)^2}\right)$.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Shared channel, Contention resolution, Throughput, Energy consumption, Randomized algorithms, Lower bound

Digital Object Identifier 10.4230/LIPIcs.DISC.2022.17

Funding *Dariusz R. Kowalski*: partially supported by the National Science Foundation grant No. 2131538 and the Polish National Science Center (NCN) grant UMO-2017/25/B/ST6/02553.

1 Introduction

A shared channel, or a multiple access channel, is one of the fundamental communication models: it allows many autonomous computing entities to communicate over a shared medium and the main challenge is how to efficiently resolve collisions occurring when more than one entity attempts to access the channel at the same time (*c.f.*, the surveys by Gallager [18] and Chlebus [7]).

In this paper we consider a classical setting, formally described as follows (c.f., [6]).[1] A *potentially unbounded* number of stations is attached to a shared channel, each of them possessing a packet that can be transmitted in a single time slot. The stations are anonymous in that they do not have any identification label (ID).[2] The computation is decentralized: every station acts independently by means of its own distributed protocol.

The communication proceeds in synchronous rounds. In each round a station can either transmit a message or listen. A transmission is successful in a given round if and only if exactly one station is transmitting in this round. In this case, the message is delivered to all stations currently active on the channel. If more than one station transmits in the same round we say that a *collision* occurs: the transmitted messages interfere each other and retransmission is necessary. Messages could be either original packets or constant-size control messages.[3]

Contention resolution problem. Each station wakes up at arbitrary time with a single packet, and in each computation prefix the number of awaken stations is finite; we will often denote it by parameter k. A station may terminate and leave the system (switches off) after its packet has been successfully transmitted. When a station switches off, it disconnects permanently from the channel. A station already awaken but not switched off yet, will be called *active*.

A *contention resolution* algorithm is a distributed algorithm that schedules the transmissions for each participating station, guaranteeing that every active station eventually transmits successfully. In other words, for each station the algorithm has to guarantee that there exists a round at which that station transmits individually, *i.e.*, without interfering with other stations.

No collision detection. Our setting is *without collision detection*. This means that in case of collision the stations do not perceive any special signal, so it will be impossible for them to distinguish between the case when more stations transmit simultaneously and the case of a silent round where no station transmits. The only feedback an active station can sense is when some station itself transmits successfully, in which case all active stations receive the transmitted message. This assumption, also called a (system) acknowledgement, is quite common in the literature (see *e.g.* [1, 3, 2]), and well motivated by technological applications such as CSMA/CA [4].

Dynamic scenario. The literature on the contention resolution problem started in the 70's and mostly considered the (simplified) *static* situation in which the stations are all activated at the same time, and thus start simultaneously their protocols, or that the activation times are based on statistical or adversarial-queueing models.

Continuing on a more recent line of research (*cf* [28, 17, 2, 3, 37, 36, 36, 35, 12, 11, 34]), in this paper we consider a general and realistic *dynamic* scenario, in which the stations get awake at arbitrary times and the sequence of activation times is totally determined by a worst-case adaptive adversary. Since a station can start its local execution of the protocol only after it has been woken up, there is no synchronization among the protocols. This makes the problem of designing a distributed algorithm considerably more challenging with respect to its simplified static counterpart.

---

[1] This setting was also used for modelling and analysis of CSMA/CA technology, c.f., [4].

[2] We assign names to the stations only for the purpose of distinguishing them in the analysis of correctness and performance.

[3] CSMA/CA and many other wireless technologies assume small-size control messages.

Local clock. Although the communication proceeds in synchronous rounds with the clocks of all the stations ticking at the same rate, our model does not allow any global clock. Each station can measure time only on the basis of its own *local clock*, which starts in the round at which the station wakes up and therefore, in the dynamic scenario considered in this paper, is not synchronized with the other clocks. We conventionally assume that a station is activated at round 0 of its local clock and can start transmitting since round 1.

Algorithmic solutions. Since the stations are anonymous, they are identical from a deterministic perspective; consequently, the only feasible solutions must be randomized. We consider *randomized* distributed algorithms for the contention resolution problem: every woken-up station has to transmit successfully its packet, regardless of its (adversarial) activation time.

All our asymptotic bounds are to be understood as high probability bounds, that is, they hold *with high probability* (in short: *whp*). We say that an event for an algorithm holds whp(k), when for a predefined parameter $\eta > 0$, the parameters of the algorithm can be chosen, so that for any k the event holds with probability at least $1 - 1/k^{\eta}$. In the intermediate steps of analysis, we will sometimes need to use the notion of whp not only with respect to the pre-assumed parameter η; in such case we will say more specifically that an event occurs "whp $1 - 1/k^{\lambda}$", for some $\lambda > 0$. Parameter λ will typically be slightly higher than η, so that at the end we could get the final result with the sought probability at least $1 - 1/k^{\eta}$.

Complexity measures. In this paper we measure the efficiency of the algorithms both in terms of *throughput* and *energy consumption*. The *throughput* is defined as follows. Given any time $t > 0$, if $n[t]$ denotes the number of stations activated up to t and $r[t]$ the number of *active rounds*, that is the rounds at which at least one station is still in the system (*i.e.,* not yet switched-off as a result of successful transmission), then the throughput is defined as the ratio $n[t]/r[t]$. Again, the goal of an algorithm will be to maximize the throughput. The ultimate goal is to achieve a *constant throughput*, *i.e.,* to show that for any time round t, the number of rounds with at least a station still active is at most a constant factor higher than the number of stations activated until time t.

Finally, concerning the *energy consumption*, we evaluate the algorithm's efficiency in terms of the maximum, over all activated stations, number of transmissions performed by a single station.

1.1 Previous work and our contribution

Contention resolution on a shared channel is a classical problem in distributed computing that is getting a lot of attention recently. The first theoretical papers date back to the 70's and considered mainly deterministic solutions for the static scenario. Below we summarize the results most relevant to ours.

Deterministic algorithms. Capetanakis [6], Hayes [22], and Tsybakov and Mikhailov [40] independently presented a deterministic tree algorithm for conflict resolution in the model with collision detection accomplishing the task in $O(k + k \log(n/k))$ rounds, for every k and n. Surprisingly, if k (or a linear upper bound on it) is given *a priori* to the stations, then the same $O(k + k \log(n/k))$ bound can be achieved even non-adaptively, without collision detection, in simple channels with acknowledgments [25]. The proof is non-constructive; later Kowalski [26] showed a more constructive solution, based on selectors (cf., [14, 23]), reaching the same asymptotic bound. Clementi, Monti, and Silvestri [16] showed a matching lower

bound of $\Omega(k \log(n/k))$, which also holds for adaptive algorithms. If collision detection is available, there is an almost matching $\Omega(k \log n / \log k)$ lower bound (also valid for adaptive algorithms) demonstrated by Greenberg and Winograd [21]. All of the above results hold for the static scenario. In the dynamic scenario, De Marco and Kowalski [17] showed that $O(k \log n \log \log n)$ time rounds are sufficient to each station to transmit successfully with a nonadaptive deterministic algorithm. Interestingly, this almost matches the $\Omega(k \log n / \log k)$ lower bound in [21], although the latter holds in a much stronger setting: for adaptive algorithms, in the static scenario and with collision detection.

Randomized algorithms. As for randomized solutions, Greenberg, Flajolet and Ladner [19] and Greenberg and Ladner [20] presented an algorithm with collision detection working in $2.14k + O(\log k)$ rounds with high probability without any *a priori* knowledge of the number k of contenders. More recently, Fernández Anta, Mosteiro and Ramon Muñoz [1] obtained the same asymptotic (optimal) bound in the model without collision detection with a non-adaptive algorithm that also ignores any knowledge about contention size k. This shows that in the static model, *i.e.,* when all the packets arrive at the same time, there is no asymptotic difference in the time complexity between adaptiveness and non-adaptiveness, even in the absence of any knowledge about channel contention.

In the dynamic scenario considered in this paper, Bender et al. [2] designed an adaptive algorithm *with collision detection* that, without any given bound on parameter k, exhibits constant throughput, linear latency and $O(\log \log^* k)$ expected transmissions per station. Later, De Marco and Stachowiak [28] proved that constant throughput and linear latency can also be achieved, with high probability, even in the more severe setting *without collision detection*, although at the expenses of a higher energy cost. In a recent breakthrough, Bender et al. [3] improved the energy cost to $O(\log^2 k)$, while preserving both constant throughput and linear latency. They considered a more restricted setting where each station is obliged to leave the system once it broadcasts its message.

Related work. The contention resolution problem has been also studied in the more general framework of multi-hop radio networks, particularly in the context of problems such as (multi-)broadcast, gossip and others in the so-called blindfold model, *i.e.* in total absence of knowledge about topology and network parameters [9, 14, 29].

Developments where similar issues on selecting stations (included broadcasting in multi-hop radio networks) under many assumptions, mainly regarding knowledge and synchrony, can be found in [13, 12, 8, 10, 15, 17, 34, 33, 38, 37, 31, 32, 30].

Our contribution. In Section 2, we design a randomized adaptive algorithm which exhibits constant throughput and only a logarithmic (in the number of participating stations k, initially unknown to participants) energy cost. Our result holds with high probability in the number of participants and this number is unknown and potentially unbounded. The analysis is made against an adaptive adversary, c.f., Theorem 7. Exploiting the adaptivity of the algorithm, we allow the stations to stay in the system even after their successful transmission. Hence, our result improves on the polylogarithmic energy cost showed in [3] if stations are not obliged to switch off once they successfully transmit their packet, but can stay in the system communicating coordination information to the other stations.

Note that even if stations are allowed to stay in the system after their successful transmissions, they contribute to the throughput and energy cost; thus, in order to optimize these measures, we have to limit such stay and additional communication to absolute minimum, which is asymptotically negligible.

In Section 3 we show that any *non-adaptive* algorithm in the model with anonymous stations cannot achieve simultaneously throughput $\Omega\left(\frac{1}{\text{polylog}(k)}\right)$ whp and energy $O\left(\frac{\log^2 k}{(\log\log k)^2}\right)$ even with a constant probability, c.f., Theorem 8. Thus, they need an energy cost that is worse than that of our algorithm by an $\Omega\left(\frac{\log k}{(\log\log k)^2}\right)$ factor for every algorithm with throughput $\Omega\left(\frac{1}{\text{polylog}(k)}\right)$. Non-adaptive protocols are such that each station chooses a distribution of transmission rounds by itself without taking into account feedback from the channel. Such protocols do not assume randomly independent choices in rounds, therefore they include a wide class of algorithms such as backoff.

Our algorithmic approach. When designing a contention resolution algorithm for the *dynamic scenario*, one has to deal with the problem caused by new arrivals of stations that, being out of sync with stations activated earlier, can interfere with their transmissions producing collisions. This interaction between new and old stations plays a major role in any contention resolution algorithm and represents the most challenging obstacle. This is made even more difficult if one has to save the number of transmissions per station, so to keep the energy cost low.

In [3] this issue is overcome in an elegant way by allowing the older players to jam the new players, so avoiding interference from the newcomers. This is done by a clever interaction between the probabilities of real transmissions and the probabilities of jamming.

Our algorithm avoids the interference between old and new stations by keeping them in two separated groups that are coordinated by means of a leader that periodically sends information about the status of the system. There are many challenges that have been tackled by our solution.

One is to assure that a leader sends information periodically, while keeping the energy cost under a logarithmic threshold.

Another one comes from the fact that the adversary can partition the execution of the algorithm in several disjoint *activity intervals*, each of them characterized by its own set of contending stations. Estimating the *total* throughput of the whole execution required the development of a new technical tool (see the notion of *random variable condensed into a vector* in Section 2.2.2) for extending the analysis of throughput for a single activity interval (with results holding with high probability with respect to the contention of the single interval, *i.e.* the number of stations involved only in that interval) to the throughput for the union of all disjoint activity intervals (with results holding with high probability with respect to the total contention, *i.e.* the total number of participating stations during the whole execution).

Additionally, our approach involves several techniques for leader election, size approximation of the participating stations, testing efficiently whether a contention resolution has been accomplished. This also highlights interesting relationships between contention resolution and other classical problems in distributed computing.

Our lower bound approach. We construct and analyze different random wake-up patterns to prove that any correct non-adaptive contention resolution algorithm that wants to be efficient in terms of throughput has an energy cost $\Omega(\log^2 k/(\log\log k)^2)$.

We start from a first weaker lower bound $\Omega(\log k/\log\log k)$, which is guaranteed even for simple wake-up times uniformly distributed, and then we square this bound by building some more complex random wake-up instances.

More precisely, we first show that if the sum of transmission probabilities is above a logarithmic threshold in all rounds of an interval, then there are small chances of having a successful transmission in that interval. Then we show how to define random wake-up

patterns such that if the algorithm wants to keep a throughput $\Omega(\frac{1}{\log^a k})$, then it has to transmit as much as to keep the sum of transmission probabilities above the logarithmic threshold in the first $\frac{k}{\text{polylog}(k)}$ rounds. Therefore for some wake-up instances, $k - \frac{k}{\text{polylog}(k)}$ nodes already incur an energy cost $\Omega(\frac{\log k}{\log\log k})$.

Then, we could recursively "pump-up" the energy cost by recursively repeating the construction for the remaining $\frac{k}{\text{polylog}(k)}$ nodes.

1.2 Conventions and notation

By convention, we assume that a station is activated at round 0 of its local clock and can start transmitting from its local round number 1. At each round a station can decide the probability of transmission by means of a randomized algorithm. Since we are dealing with adaptive algorithms, these probabilities may depend on the history of the channel feedback and do not have to be independent over rounds.

Although there is no global time accessible to the stations, in the analysis we will need a *reference clock* (not visible to the stations) that allows us to argue about the behaviour of all the stations involved in the computation at a given moment.

For any time t of a given reference clock, we denote by $\hat{A}[t]$ the set of stations activated until time t. The transmission probability assigned by the protocol to a station $v \in \hat{A}[t]$ at time t will be denoted by $q_v[t]$. Some already activated stations, however, may not be active during the protocol execution in time t, because of switching off earlier; therefore, we use $A[t] \subseteq \hat{A}[t]$ to denote the set of stations that are *still active* at time t.

We define the *sum of transmission probabilities* at time t as follows: $\sigma[t] = \sum_{u \in A[t]} q_u[t]$.

Analogously, we will also need to consider the above sum over all activated stations until time t (*i.e.* not considering switches-off): $\hat{\sigma}[t] = \sum_{u \in \hat{A}[t]} q_u[t]$.
Surely, if t' is the time of the first successful transmission, then $\hat{\sigma}[t] = \sigma[t]$ for every $t < t'$.

In our analysis we will also need to deal with the sum of probabilities used by a station up to some time of its local clock. We define the *sum of transmission probabilities of an arbitrary station* up to local time i as: $s(i) = \sum_{j=1}^{i} p(j)$, where $p(j)$ denotes the transmission probability of the station at round j of its *local clock*. We do not need to specify the station which this probabilities refers to, as it will be clear from the context.

2 An adaptive algorithm for unknown contention

We now describe our protocol `AdaptiveCTLE`, which resolves the contention with constant throughput and logarithmic energy, without any knowledge on the number of contenders. The number of contenders could be arbitrary and they could be activated during an arbitrarily long time interval. Besides the data packet itself, each station can send a one-bit control message. For the sake of presentation we will refer to these control messages as <`D mode`> (encoded with bit 0) and <`any D-station left?`> (encoded with bit 1).

High-level description. The reader can refer to Figure 1 for a graphical representation of algorithm's behaviour. The algorithm alternates between two modes: a *leader election mode* (*L mode*) and a *dissemination mode* (*D mode*). The first mode aims at getting a synchronized subset of stations and electing a leader, which has the task of coordinating the computation in the next dissemination mode (it will also help in Protocol `Estimate&Increase`, which is a sub-routine executed in *D* mode). This is actually the contention resolution among the synchronized subset of stations defined in the preceding execution of the leader election

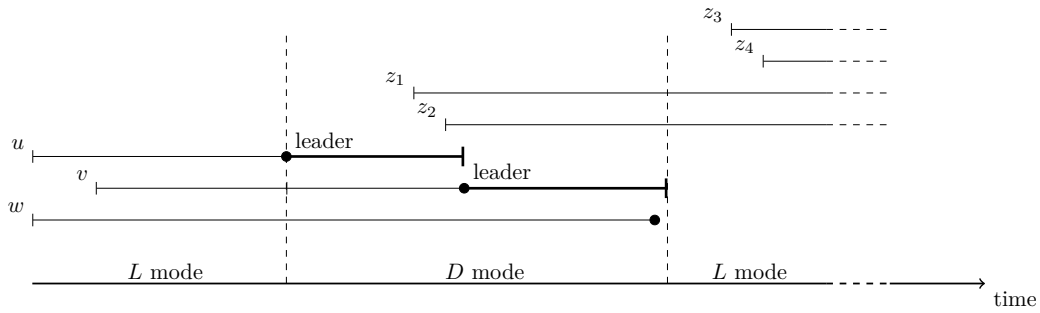

Figure 1 Horizontal segments represent the activity periods of stations, black circles indicate *first* successful transmissions for each station. Stations u, v and w start in L mode. Then, u is elected a leader: the stations are now synchronized and the D mode starts. Thicker segments show the change of leadership among stations every $O(\log k)$ rounds. Stations z_1 and z_2 wake up while u, v and w are in D mode: they remain *pending* until the current D mode is active (see the middle interval on the picture). Once u, v and w have switched off, z_1 and z_2 start a new L mode. Other stations (z_3 and z_4) can possibly join in arbitrary times. Once a leader has been elected a new D-mode starts and so forth. This process is iterated until a D mode ends and no station wakes up during its execution.

mode. The leader does not remain the same for the whole execution of the algorithm, but the stations take turns in this role, in order to keep the maximum number of transmissions below a logarithmic threshold (this will be assured by Protocol `SlowIncrease`).

All the stations running the D mode use a synchronized clock ticking rounds modulo 4. Odd rounds are used to execute the actual contention resolution protocol, while even rounds will be used together with the leader to learn and send additional information to the participating stations.

At any time, the system is either in L mode or in D mode. A station involved in the L mode (resp., D mode) will be called an L-station (resp., a D-station). A newly awaken station learns the mode of the system during the first 4 rounds. Then it remains with an empty status until it keeps receiving, once every 4 rounds, a message <`D mode`> from the current leader. This message informs the newcomers that the system is busy with the dissemination mode. We call these stations with an empty status *pending stations*. This status of "pending" will last as long as the stations that are currently in D mode have not switched-off. It is the leader that will inform when this happens by ceasing to send the <`D mode`> message and switching off. When this happens, all the pending stations start a new L mode, and so forth. This process is iterated until no new station is injected in the system; more precisely, when, after the switching-off of all the current D-stations, no pending station is waiting to start a new L mode.

A formal description of the algorithm can be found in the following pseudocodes.

2.1 Pseudocodes

The main protocols of the algorithm are `AdaptiveCTLE` (Protocol 1), which is the first protocol executed by a station when it is activated, and `AdaptiveLeader` (Protocol 2), which is executed by the leader.

Protocol 1 and 2 (`AdaptiveCTLE` and `AdaptiveLeader`). A newly awaken station u starts with the execution of Protocol `AdaptiveCTLE`. It takes the status of pending station and keeps it while in the loop in line 2 of this protocol, *i.e.,* while it keeps receiving the <`D`

Algorithm 1 `AdaptiveCTLE` (*executed by any station u*).

STATUS $\leftarrow \emptyset$
while STATUS $\notin \{L, D\}$ **do** // *a woken up station keeps waiting (pending station)*
  listen to the channel
  if u does not receive message <`D mode`> in 4 consecutive rounds **then**
    STATUS $\leftarrow L$ // *u becomes an L-station*
while u is active **do**
  if STATUS $= L$ **then**
    execute `DecreaseSlowly` // *the first successful station becomes the leader*
    if u has been elected the leader **then**
      execute `AdaptiveLeader` // *see Protocol 2*
    STATUS $\leftarrow D$ // *once a leader is elected, station switches to dissemination mode*
    `time_counter` $\leftarrow 0$ /* *now all awaken stations are synchronized and* `time_counter` *will denote the current round number started at the time the leader has been elected* */
  if STATUS $= D$ **then**
    if `time_counter` is odd **then**
      execute round $\lfloor(\texttt{time\_counter} + 1)/2\rfloor$ of `Estimate&Increase`(c) (switch-off at the first successful transmission)
    else if `time_counter` $= 2^x$, for some integer $x \geq 2$, **then**
      transmit message <`any D-station left?`>

Algorithm 2 `AdaptiveLeader` (*executed by the leader*).

1: **while** u is the leader **do**
2: **if** `time_counter` is odd **then**
3: participate as leader in `Estimate&Increase`(c)
4: **else if** `time_counter` $= 2^x$, for some $x \geq 2$, **then**
5: transmit message <`any D-station left?`>
6: **if** the message is received **then** // *if D mode has terminated*
7: switch off
8: **else** transmit <`D mode`> // *leader without acknowledgment => D mode continues*

Algorithm 3 `DecreaseSlowly` (*executed by a station u*) [24].

1: $q \leftarrow$ some constant > 0
2: $i \leftarrow 0$
3: **while** u is active **do**
4: transmit the message with probability $q \cdot \frac{1}{2q+i}$
5: **if** transmission is successful **then**
6: become a leader
7: $i \leftarrow i + 1$

Algorithm 4 `Estimate&Increase`(c) (*executed by a station u*).

1: **while** `Test` is true **do**
2: Execute `Estimate` to obtain an estimate value k
3: Execute `SlowIncrease`(k, c)

mode> message once every 4 rounds. If within 4 consecutive rounds of waiting, station u does not get the <D mode> message, then the system is not in D mode and three cases may have occurred:

(a) there was no active station when u has been woken up;
(b) the system was in L mode when u has been woken up;
(c) a D mode was running when u has been woken up, but all D-stations delivered their messages and switched off.

In all three cases, the station exits the loop in line 2 as an L-station and starts the subsequent loop. Being an L-station, it starts executing protocol `DecreaseSlowly` (cf. the pseudocode of Protocol 3). This is a wake-up protocol introduced in [24] whose goal is to get just one successful transmission among an asynchronized set of stations.

Once a successful transmission appears in some round t, the station which transmitted in t becomes the *leader*. At this point the leader and all other stations that were alive at round t, are synchronized. They set up a new variable `time_counter` to 0 and a D mode starts: the leader starts the execution of Protocol `AdaptiveLeader`, while the other synchronized stations continue with the execution of `AdaptiveCTLE`.

Let us denote by C such a *synchronized* subset of stations (not including the leader). We can assume that a global clock (represented by variable `time_counter` initiated by the leader) starts for all the stations in C at the round in which the leader was elected. This allows us to use a contention resolution protocol for a synchronized set of stations. We accomplish this task with Protocol 4, `Estimate&Increase`, which guarantees that all synchronized stations in C transmit successfully within $O(|C|)$ rounds after the synchronization round whp in $|C|$. During an execution of `Estimate&Increase`, all stations from set C switch off directly after a successful transmission. In order for such a protocol to work properly, it is necessary to avoid that stations that have woken up during its execution (newcomers) could interfere disturbing the transmissions.

In order to make it possible that the newcomers understand what is happening in the system, the algorithm `Estimate&Increase` is executed in odd rounds only, while even rounds are devoted to performing the following kind of coordination. In rounds $t = 2^x$, for some integer $x \geq 2$, the D-stations and the leader send message <any D-station left?>, as stated in lines 17 and 5 of their respective Protocols 1 and 2 (note that this requires a number of transmissions that is only logarithmic in the length of protocol `Estimate&Increase`, that is $O(\log |C|)$).

This message will be successfully heard by the leader (and delivered to the pending stations busy in the loop of line 2) if and only if the leader is the only transmitter, that is, when all the D-stations in set C have switched off. If the transmission is successful, the leader switches off immediately. If, on the other hand, this message is not successfully acknowledged, then it means that there is still some D-station active in the system. In this case, in all subsequent even rounds that are not reserved to message <any D-station left?>, the leader sends message <D mode> informing the pending stations that the D mode has not terminated, and so they have to keep waiting (*i.e.,* listening) in the loop.

We can now see how the above strategy guarantees that any station activated during the execution of protocol `Estimate&Increase`, remains pending in the loop of line 2 silently waiting for its termination and then it exits the loop as an L-station. Indeed, while `Estimate&Increase` is running, the leader transmits a <D mode> message once every at most 4 rounds (all even rounds with the exception of rounds 2^x, for $x \geq 2$). This keeps the station waiting in the loop. Once, the D-mode has terminated, the pending station hears no <D mode> message during 4 consecutive rounds and therefore it exits the loop as an L-station, starting a new L-mode. This process is iterated until it happens that no pending station is waiting in the loop of line 2, *i.e.* when no other station is injected in the system.

Protocol 3 (`DecreaseSlowly`). This protocol is used to elect a leader among the set of *asynchronized* stations running in L mode. For this task we use a wakeup protocol introduced in [24]. This protocol let just one station to successfully transmit. This station takes the role of a leader.

Protocol 4 (`Estimate&Increase`). The purpose of this protocol is to solve the contention among the *synchronized* set of stations running in D mode. The leader will also participate to the computation (as stated on line 3 of Protocol `AdaptiveLeader`). The purpose is accomplished by means of a sequence of two protocols: first, `Estimate`, which computes a 2-approximation of the number of synchronized contenders, and subsequently, `SlowIncrease`, which uses the previously computed estimate to let the stations to successfully transmit their messages and switch off. The two protocols `Estimate` and `SlowIncrease` will be executed repeatedly until all the stations have switched off. This condition is checked by a function `Test`. Let us now describe in more details the behavior of each of these three ingredients.

Protocol `Estimate`. In order to compute our 2-approximation of the number of contenders, we adapt Algorithm $(1+\epsilon)$-`approximation` for the beeping model [5] to our shared channel without collision detection. Because in our model beeps are not explicitly recognized by the channel, we emulate each round r of Algorithm $(1+\epsilon)$-`approximation` by the following two `echo` rounds (see [27]) in our model:

- round $2r$ of Protocol `Estimate`: if a participating station is scheduled to transmit in the corresponding round r of $(1+\epsilon)$-`approximation`, it also transmits in round $2r$ of `Estimate`;
- round $2r+1$ of Protocol `Estimate`: if a participating station is scheduled to transmit in the corresponding round r of $(1+\epsilon)$-`approximation`, it also transmits in round $2r+1$ and the leader transmits as well.

If nothing is heard in the first `echo` round and the leader is heard in the second `echo` round, then it means there is no beep in round r of $(1+\epsilon)$-`approximation`; otherwise there is a beep.

Protocol `SlowIncrease`(k, c). Once we have obtained an approximate estimate of the number of participating stations, we can use an algorithm for contention resolution which exploits such an information. For this task we can use protocol `NonAdaptiveWithK` (k, c) [28], which uses an upper bound of the number k of contenders and a sufficiently large constant c (which determines the probability of success). Each successfully transmitting station automatically swaps the leadership role with the current leader, which switches off.

In order to assure logarithmic energy of the leader, if there is no swap in consecutive $\log k$ rounds of the original protocol `NonAdaptiveWithK` (k, c) [28], the leader runs a tournament that elects another leader in $O(\log k')$ rounds, where k' is the actual number of participants. The binary search with `echo` protocol from [27] is used, coordinated by the leader, in the beginning of which stations choose random ids from interval $[1, O(k)]$. It is possible that more than one station selects the same id, in which case the leader discovers an echo for that value, and runs the tournament again but this time only for the colliding stations. If the time of the tournament exceeds $2 \log k$, its actual length divided by 2 is set as the power of 2 in the new estimate of k – the number of participating stations. Note that the time, and thus also the energy, of the tournament is amortized by the value of $\log k$ of the estimate, and so the following number of rounds in which the original protocol `NonAdaptiveWithK` (k, c) is executed.

Function `Test`. Similarly as in the protocol `Estimate`, the same `echo` procedure (a single 2-round execution of it) is used with respect to the remaining participants of the superior protocol and the current leader. If there is a beep, the test is true, otherwise it returns false.

2.2 Analysis of throughput and energy

Correctness and time complexity of our algorithm `AdaptiveCTLE` will be finally proved in Theorem 7. We will go through the following steps.

First, we analyze algorithm `AdaptiveCTLE` in an *activity interval*, *i.e.* a maximal size interval of consecutive active rounds. In doing so, we start from the analysis of the basic sub-routine protocols (Section 2.2.1), and then put them together into the analysis of the whole activity interval (Section 2.2.2).

Next, we put the disjoint activity intervals together, using their independence and partitioning the activity of the adversary into classes of sub-adversaries, one for each activity interval (Section 2.2.3).

2.2.1 Analysis of protocols

Consider an activity interval with k contenders. The first step is to analyze the performance of procedure `DecreaseSlowly`. In [24], an algorithm *Decrease Slowly* has been presented for the first time, and it was proven to solve the wake-up problem (*i.e.*, allow just one successful transmission) in $O(k \log k)$ rounds whp(k). In [28] it was improved to $O(k)$ rounds whp(k). Both analysis assumed non-adaptive adversary, *i.e.*, an adversary scheduling the wake-up times in advance. In the following lemma, we show more detailed properties of this algorithm, under a stronger adaptive adversary (as considered in this paper). The proof is deferred to the appendix.

▶ **Lemma 1.** *Algorithm* `DecreaseSlowly` *finishes wake-up in $O(k)$ rounds and with $O(\log k)$ energy whp(k), where k is the number of activated stations during the activity interval of this execution. Moreover, for any $k' > k$, algorithm* `DecreaseSlowly` *finishes wake-up in $O(k \log(k'/k))$ rounds and with $O(\log k')$ energy whp(k').*

The two ingredient protocols of algorithm `Estimate&Increase` satisfy the following:

▶ **Lemma 2** ([5])**.** *Protocol* `Estimate` *outputs a 2-approximation of the number k of stations in D-mode in $O(\log k)$ rounds and $O(\log k)$ energy whp(k), and for any $k' > k$, it outputs a 2-approximation of the number k of stations in D-mode in $O(\log k')$ rounds and energy whp(k').*

▶ **Lemma 3** ([28])**.** *Protocol* `SlowIncrease`(k, c) *solves the contention resolution problem for at most k stations in D-mode in $O(k)$ rounds and with $O(\log k)$ energy whp(k).*

▶ **Lemma 4.** *Algorithm* `Estimate&Increase`(c) *solves the contention resolution problem within an activity interval in $O(k)$ rounds and with $O(\log k)$ energy whp(k), where k is the number of D-stations in the beginning of this execution. Moreover, for any $k' > k$, algorithm* `Estimate&Increase`(c) *solves the contention resolution problem within an activity interval in $O(k \log(k'/k))$ rounds and with $O(\log k')$ energy whp(k').*

Proof. The first part follows from putting together Lemmas 2 and 3. The second part follows from repeating the above independently $O(\log(k'/k))$ times, until the `Test` becomes false and the algorithm stops. The standard probabilistic argument applies here because of two reasons: these repeating parts are synchronized by the `Test` run in the beginning of the loop, and stations participate in only a single iteration of the loop.

In the Protocol `Estimate&Increase` and the associated confirmation rounds, there is a constant number of transmissions per one successful transmission, on average, thus $O(\log k)$ whp(k) and $O(\log k')$ whp(k') in an execution of length $O(k \log(k'/k))$. ◀

2.2.2 Analysis of a single activity interval

In order to show that the performance guarantees of our algorithm hold with high probability with respect to the total number of stations awaken during any execution, independently of how the execution is partitioned into disjoint activity intervals, we introduce the notion of condensed random variables. Next, in Lemma 5, whose proof is in the appendix, we use such a tool to show that the sum of the lengths of the activity intervals is bounded with high probability with respect to the total number of stations awaken.

Let $c > 0$ be a sufficiently large constant, depending on the exponent η in the formula for whp(). We say that positive integer random variables ℓ_i, for $1 \leq i \leq x$, are *condensed* into a vector $\kappa = (w_1, \ldots, w_y)$, for some positive integer y such that $w_1 \leq \ldots \leq w_y = x$, if for every $1 \leq j \leq y$, set $L_j = \{i : E[\ell_i] = O(2^j)$ & $\ell_i \leq c \cdot 2^j$ holds whp(2^j) & $\ell_i \leq c \cdot 2^j \log(\bar{\kappa}/2^j)$ holds whp$(\bar{\kappa})\}$ is of size w_j, where $\bar{\kappa} = c \cdot \sum_{j=1}^{y} 2^j w_j$.

The following technical fact holds (see the appendix for the proof).

▶ **Lemma 5.** *Let ℓ_i, for $1 \leq i \leq x$, be random variables condensed into $\kappa = (w_1, \ldots, w_y)$, for some positive integer y. Then, $\sum_{i=1}^{x} \ell_i = O(\bar{\kappa})$ whp$(\bar{\kappa})$.*

We now show the performance guarantees of our algorithm within each activity interval.

▶ **Lemma 6.** *Algorithm* `AdaptiveCTLE` *solves the contention resolution problem within an activity interval in $O(k)$ rounds and with $O(\log k)$ energy whp(k), where k is the total number of stations activated during this execution. Moreover, for any $k' > k$, algorithm* `AdaptiveCTLE` *solves the contention resolution problem within an activity interval in $O(k \log k')$ rounds and with $O(\log k')$ energy whp(k').*

Proof. Let I be the interval of rounds involved in this execution of `AdaptiveCTLE`. The system starts in L mode (apart from an initial waiting period of at most 4 rounds spent in the first `while` loop of Protocol `AdaptiveCTLE`, during which the newly activated stations, as pending stations, realize that the system was inactive) and then it enters the D mode.

The execution of the algorithm is composed of a sequence of L mode/D mode executions, in such a way that the ith L mode is followed by the ith D mode. Each D mode execution involves the same set of stations that participated to the previous L mode, and the ith L mode execution, for $i > 1$, involves the stations that were pending in the previous D mode. This will continue until the end of interval I which occurs when all the D-stations of the last execution of `Estimate&Increase` switch off and there is no station pending at the end of it.

Let x be the number of such L mode/D mode executions. For $1 \leq i \leq x$, let ℓ_i^L be the length of the ith L mode and ℓ_i^D be the length of the following D mode. By Lemma 1 we have that $\ell_i^L = O(m_i)$ whp(m_i), where m_i is the number of stations that participated in the ith L mode. This is also the number of stations involved in the next D mode. By Lemma 4, it follows that (1) $\ell_i^D = O(m_i)$ whp(m_i), (2) $E[\ell_i^D] = O(m_i)$ and (3) $\ell_i^D = O(m_i \log(k/p_i))$. Hence, considering the lengths $\ell_i^{LD} = \ell_i^L + \ell_i^D$ of each L mode/D mode execution, we also have (1) $\ell_i^{LD} = O(m_i)$ whp(m_i), (2) $E[\ell_i^{LD}] = O(m_i)$ and (3) $\ell_i^{LD} = O(m_i \log(k/p_i))$.

Let w_j be the number of executions i such that $m_i \leq c \cdot 2^j$. It follows that the random variables ℓ_i^{LD}, for $1 \leq i \leq x$, are condensed into $(w_1, w_2, \ldots, w_y)$. By Lemma 5 we have that $\ell = \sum_{j=1}^{x} \ell_j^{LD} = O(\sum_{j=1}^{x} m_j) = O(k)$ whp(k).

Lemmas 1 and 4 guarantee $O(\log k)$ energy whp(k) for the stations except the activity of the leader in Protocol 1. It is easy to see that the transmissions of the leader could continue for a large period of time, which could go over the desired $O(\log k)$ upper bound. This is due to the total execution time of Protocol `Estimate&Increase` called on line 15, during which the leader keeps sending messages (along with the other synchronized stations) until it gets an acknowledgement. However, the algorithm resolves this issue by requesting the stations participating to `Estimate&Increase` swapping the role of leader in such a way that none of them transmits for more than $O(\log k)$ rounds, see description of Protocol `SlowIncrease`(k, c). The second part of the lemma follows directly by applying second parts of the results for the ingredient protocols (for any $k' > k$), in Lemmas 1 and 4, and estimating each $\log(k'/p_i)$ from above by $\log k'$. ◀

2.2.3 Putting activity intervals together

Finally, we are ready for the main theorem of this section. We will be using Lemma 5 again, where the ℓ_i's correspond now to the lengths of subsequent activity intervals and $\bar{\kappa}$ is an upper linear estimate of the activated stations in the whole considered execution.

▶ **Theorem 7.** *Algorithm* `AdaptiveCTLE` *has constant throughput and* $O(\log k')$ *energy,* whp(k')*, where* k' *is the total number of awaken station in a considered prefix of the execution of Algorithm* `AdaptiveCTLE`*.*

Proof. Fix an arbitrary execution of the algorithm. For any time round t, let $Q[t]$ be the set of rounds $r \leq t$ at which there is at least a station still active and $n[t]$ be the total number of stations activated until time t. We need to show that the size of $Q[t]$ is at most a constant factor higher than $n[t]$.

Depending on the distance between consecutive activation times (that are controlled by the adversary and can be arbitrarily large) the execution of the algorithm can be split into several independent executions involving disjoint subsets of stations. At the end of each execution, all the stations awaken during it will be switched-off. The time rounds between two consecutive executions do not belong to $Q[t]$.

Therefore, there exists an integer $1 \leq x \leq k$ such that $Q[t]$ can be partitioned into disjoint time intervals $I_1, \ldots, I_x$, each of them corresponding to an independent execution of the algorithm on a set S_i of stations, where all these subsets form a partition of the set of all the stations activated until time t, formally $\bigcup_{1 \leq i \leq x} S_i = \hat{A}[t]$ and $S_i \cap S_j = \emptyset$ for $i \neq j$. We apply Lemma 5 to these intervals in exactly the same way as we applied it to the independent executions of L mode and `Estimate&Increase` inside activity interval in the first part of the proof of Lemma 6. Now, the base properties of the lengths of activity intervals are guaranteed by Lemma 6, where k' stands for the total number of awaken stations in the considered prefix of execution, and Lemma 5 implies the theorem for k' being upper bounded by $\bar{\kappa}$ with respect to a constant factor.

Finally, note that due to the independence of the activity intervals, each awaken station participates in only one of them, therefore each station's energy is $O(\log k')$ whp(k'), by Lemma 6 and the union bound over the participating k' stations. ◀

3 A trade-off between throughput and energy of non-adaptive algorithms

An arbitrary sequence of k activation times for k stations will be called an *instance of at most k stations* and denoted by $I(k)$. For any instance $I(k)$, we will use a reference clock starting when the first station is activated. All the following rounds t refer to this clock.

Given an algorithm $\mathcal{A}$, we let $T_{\mathcal{A}}(I(k))$ be the maximum time needed for $\mathcal{A}$ to assure a successful transmission of any station in instance $I(k)$ whp. Analogously, we let $E_{\mathcal{A}}(I(k))$ be the expected number of transmissions per station spent by algorithm $\mathcal{A}$ on instance $I(k)$.

▶ **Theorem 8.** *Let $\tau(x) = \Omega(1/(\log^a x))$ for any constant $a > 0$. There exists an instance $I(k)$ such that any non-adaptive algorithm not knowing k and achieving throughput $\tau(k)$ whp, requires $\Omega(\log^2 k/(\log\log k)^2)$ expected transmissions per station.*

Proof of Theorem 8. In order to prove the theorem, from now on we fix an arbitrary non-adaptive algorithm $\mathcal{A}$ achieving throughput $\tau(k) = \Omega(1/(\log^a x))$ whp. All the following results are meant to hold for such an algorithm $\mathcal{A}$. Also, to simplify the description, all bounds involving throughput are meant to hold with high probability even when not explicitly stated. Moreover, we denote by $T(k)$ (resp. $E(k)$) the maximum $T_{\mathcal{A}}(I(k))$ (resp. $E_{\mathcal{A}}(I(k))$) taken over all instances $I(k)$ activating k stations.

▶ **Fact 9.** $T(k) \leq k/\tau(k)$.

Proof. Suppose on the contrary that $T(k) > k/\tau(k)$. Then for $t = T(k)$, the ratio between the number of activated stations and the number of active rounds will be

$$\frac{n[t]}{r[t]} \leq \frac{k}{T(k)} < \frac{k \cdot \tau(k)}{k} = \tau(k),$$

which contradicts the assumption on the throughput of algorithm $\mathcal{A}$. ◀

We start with the following lemma showing a first lower bound on the average number of transmissions. We will improve such a bound later on.

▶ **Lemma 10.** $E(k) = \Omega(\log k/\log\log k)$.

Proof. Let us build a random instance of k stations as follows. By the hypothesis on throughput and Fact 9, we can assume that $T(k) \leq k/\tau(k)$ whp. Let v be a station activated at time 1 of the instance. By definition of $T(k)$, v has to transmits successfully within $T(k)$ rounds whp. The number of rounds at which v transmits in the time period $[1, T(k)]$ is a random variable X such that

$$E(X) = E(k) = s(T(k)) = \sum_{i \in [1, T(k)]} p(i) \ .$$

By Markov's inequality we have

$$\Pr\left(X < 2E(X)\right) > 1/2 \ . \tag{1}$$

We now let the activation times of the other $k-1$ stations be distributed uniformly at random among the $T(k)$ rounds. Each station transmits with probability $p(1)$ at the first round it switches on. Since the algorithm does not know k, the probability $p(1)$ does not depend on k. Therefore, we have that at any round of $[1, T(k)]$ in which v transmits, the probability that this transmission is not successful is the probability that any of the other $k-1$ stations transmits at the same time, that is $(k-1) \cdot p(1) \cdot (1/T(k)) = \Omega(\tau(k))$, where the asymptotic bound is due to the inequality $T(k) \leq k/\tau(k)$. Thus, the probability of no successful transmission for station v during the whole interval $[1, T(k)]$ is at least $\Omega\left(\tau(k)^X\right)$.

Hence, this probability is at most $1/k^\eta$ for any predetermined constant $\eta > 0$, only when the number of transmissions of v is $X = \Omega(\log k/\log(1/\tau(k))) = \Omega(\log k/\log\log k)$. By Equation (1), it follows that $E(X) = \Omega(\log k/\log\log k)$ and the lemma follows. ◀

Now we show that if the algorithm transmits as much to keep the sum of transmission probabilities above a logarithmic threshold, then the probability of having a successful transmission becomes very low. Notice that before the first successful transmission occurs, we have $A[t] = \hat{A}[t]$. For this reason, in the following calculations we consider bounds on $\hat{\sigma}[t]$ instead of $\sigma[t]$, as they are equivalent until the first successful transmission appears.

▶ **Lemma 11.** *Let $T \leq k^2$. There exists a constant $\gamma > 0$ such that if $\hat{\sigma}[t] \geq \gamma \log k$ for every $t \in [1, T]$, then the probability of having at least one successful transmission in the time interval $[1, T]$ is smaller than $1/k$.*

Proof. The probability of having *at least one* successful transmission in the time interval $[1, T]$, is equivalent to the probability of having the *first* transmission in any round t of this interval. For a fixed round t, this probability is at most

$$\sum_{v \in \hat{A}[t]} p(t - t_v) \prod_{w \in \hat{A}[t], w \neq v} (1 - p(t - t_w)) \leq \hat{\sigma}[t] e^{-\hat{\sigma}[t]+1} ,$$

which can be made smaller than $1/k^3$, for a sufficiently large γ. By taking the union bound over all the $T \leq k^2$ rounds of the interval, we get that this probability is at most $1/k$. ◀

The following lemma shows that the hypothesis on throughput implies that the sum of transmission probabilities will be maintained above the logarithmic threshold determined in the previous lemma. In other words, if the algorithm wants to be efficient in terms of throughput, then it has to lose in terms of energy.

▶ **Lemma 12.** *Let γ be the constant determined in Lemma 11. There exists a constant $c > 0$ and an instance $I_1(k)$ such that, for k sufficiently large, $\hat{\sigma}[t] \geq \gamma \log k$ in all rounds $t \in [1, T(k/(c \log^{1+a} k))]$.*

Proof. By the hypothesis on throughput and Fact 9 we know that $T(k) \leq O(k \log^a k)$. Hence, $T(k/\log^{1+a} k) \leq O(k/\log k)$. Consequently, for any constant $c' > 0$ there exists a constant $c > 0$ such that $T(k/(c \log^{1+a} k)) \leq k/(c' \log k)$, for k sufficiently large.

Construction of instance $I_1(k)$. Letting $c' = \gamma/p(1)$, we can construct an instance $I_1(k)$ for k contending stations as follows. In each round $t \in [1, T(k/(c \log^{1+a} k))]$ we switch on $c' \log k$ stations. Note that for this task, it is sufficient to activate at most k stations, indeed:

$$c' \log k \cdot T\left(\frac{k}{c \log^{1+a} k}\right) \leq c' \log k \cdot \frac{k}{c' \log k} = k .$$

Each station transmits with probability $p(1)$ in the round it is switched on. Therefore, in every $t \in [1, T(k/(c \log^{1+a} k))]$, $\hat{\sigma}[t] \geq c' \log k \cdot p(1) = \gamma \log k$. ◀

Now we can show that we can build an instance of k stations such that the transmissions are mainly distributed at the end of the considered interval of $T(k)$ rounds.

▶ **Lemma 13.** *For some constant c, $E(k) - E\left(\frac{k}{c \log^{1+a} k}\right) = \Omega\left(\frac{\log k}{\log \log k}\right)$.*

Proof. In order to prove the lemma, we build a corresponding instance $I_2(k)$ and analyze its properties. We start as follows. Let γ be the constant determined by Lemma 11 and take an instance $I_1(k/2)$ of $k/2$ stations as guaranteed by Lemma 12. This lemma implies that $\hat{\sigma}[t] \geq \gamma \log(k/2)$ in all rounds $t \in \left[1, T\left(\frac{(k/2)}{c' \log^{1+a}(k/2)}\right)\right]$, for some constant c'. There exists

a constant c such that $T\left(\frac{(k/2)}{c' \log^{1+a}(k/2)}\right) \geq T\left(\frac{k}{c \log^{1+a} k}\right)$. Therefore, by Lemma 11, there is no successful transmission in all rounds $1, 2, \ldots, T\left(\frac{k}{c \log^{1+a} k}\right)$ whp. Hence, we can conclude that whp a station v starting the protocol at round 1 is not able to transmit successfully in the interval $\left[1, T\left(\frac{k}{c \log^{1+a} k}\right)\right]$.

Now we continue the construction of an instance $I_2(k)$ by distributing uniformly at random the remaining $k/2$ stations in the interval $[T(k/(c \log^{1+a} k), T(k)]$. The non-adaptive algorithm $\mathcal{A}$ assigns a fixed sequence of transmissions to v in this interval. Recalling Fact 9, each of these transmissions is not successful with probability larger than $(k/2) \cdot p(1) \cdot (1/T(k)) = \Omega(\tau(k)) = \Omega(1/\log^a k)$.

Thus, in order to have a successful transmission with high probability, station v needs to transmit $\Omega(\log k / \log\log k)$ times in the interval $[T(k/(c \log^{1+a} k)), T(k)]$, as $(1/\log^a k)^{\Omega(\log k/\log\log k)} = 1/\mathrm{poly}(k)$.

Thus, analogously as in the proof of Lemma 10, in order to have a successful transmission with high probability, station v needs to transmit $\Omega(\log k / \log\log k)$ times between round $T(k/(c \log^{1+a} k))$ and $T(k)$. Therefore, $E(k) - E(k/(c \log^{1+a} k)) = \Omega(\log k / \log\log k)$. ◀

Finally, the next lemma concludes the proof of Theorem 8.

▶ **Lemma 14.** $E(k) = \Omega(\log^2 k/(\log\log k)^2)$.

Proof. We can write down a telescoping sum, where c is the constant determined in Lemma 13:

$$\begin{aligned} E(k) \;=\; & \big(E(k) - E(k/(c \log^{1+a} k))\big) + \big(E(k/(c \log^{1+a} k)) - E(k/(c \log^{1+a} k)^2)\big) + \\ & + \big(E(k/(c \log^{1+a} k)^2) - E(k/(c \log^{1+a} k)^3)\big) + \ldots \end{aligned}$$

The thesis follows by noting that this sum has $\Omega(\log k / \log\log k)$ terms, and the first half of these terms are $\Omega(\log k / \log\log k)$ by Lemma 13. ◀

4 Open problems

The most interesting open direction is to study tradeoff between energy consumption and other measures. In particular, is logarithmic energy necessary for anonymous shared channel against adaptive adversary in order to achieve constant throughput? If so, could we lower the energy requirement by allowing slightly smaller throughput, and if so, how much smaller? How the performance changes if we start restricting protocols, for instance, by limiting randomness or/and number of listening rounds, not allowing any control bits or requesting successful stations to disconnect immediately. Considering occasional failures and/or accounting rounds when stations actively listen to the energy measure are examples of other challenging directions.

References

1 A. Fernández Anta, M. A. Mosteiro, and J. Ramon Mu noz. Unbounded contention resolution in multiple-access channels. *Algorithmica*, 67:295–314, 2013.

2 M. A. Bender, T. Kopelowitz, S. Pettie, and M. Young. Contention resolution with log-logstar channel accesses. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC)*, pages 499–508, Cambridge, MA, USA, 2016. ACM.

3 Michael A. Bender, Tsvi Kopelowitz, William Kuszmaul, and Seth Pettie. Contention resolution without collision detection. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 105–118, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384305.

4 Giuseppe Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *Selected Areas in Communications, IEEE Journal on*, 18:535–547, April 2000. doi:10.1109/49.840210.

5 Philipp Brandes, Marcin Kardas, Marek Klonowski, Dominik Pająk, and Roger Wattenhofer. Fast size approximation of a radio network in beeping model. *Theoretical Computer Science*, 810:15–25, 2020. Special issue on Structural Information and Communication Complexity. doi:10.1016/j.tcs.2017.05.022.

6 J. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Transactions on Information Theory*, 25:505–515, 1979.

7 B. S. Chlebus. Randomized communication in radio networks. In P. M. Pardalos, S. Rajasekaran, J. H. Reif, and J. D. P. Rolim, editors, *Handbook on Randomized Computing*, pages 401–456. Springer, New York, NY, USA, 2001.

8 Bogdan S. Chlebus, Leszek Gąsieniec, Dariusz R. Kowalski, and Tomasz Radzik. On the wake-up problem in radio networks. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming*, pages 347–359, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

9 Bogdan S. Chlebus, Leszek Gąsieniec, Alan Gibbons, Andrzej Pelc, and Wojciech Rytter. Deterministic broadcasting in unknown radio networks. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, pages 861–870, USA, 2000. Society for Industrial and Applied Mathematics.

10 Bogdan S. Chlebus and Dariusz R. Kowalski. A better wake-up in radio networks. In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing*, PODC '04, pages 266–274, New York, NY, USA, 2004. Association for Computing Machinery. doi:10.1145/1011767.1011806.

11 Bogdan S. Chlebus, Gianluca De Marco, and Dariusz R. Kowalski. Scalable wake-up of multi-channel single-hop radio networks. *CoRR*, abs/1411.4498, 2014. arXiv:1411.4498.

12 Bogdan S. Chlebus, Gianluca De Marco, and Dariusz R. Kowalski. Scalable wake-up of multi-channel single-hop radio networks. *Theor. Comput. Sci.*, 615:23–44, 2016. doi:10.1016/j.tcs.2015.11.046.

13 Bogdan S. Chlebus, Gianluca De Marco, and Muhammed Talo. Naming a channel with beeps. *Fundam. Informaticae*, 153(3):199–219, 2017. doi:10.3233/FI-2017-1537.

14 M. Chrobak, L. Gasieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks. *Journal of Algorithms*, 43:177–189, 2002.

15 Marek Chrobak, Leszek Gasieniec, and Dariusz Kowalski. The wake-up problem in multi-hop radio networks. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 992–1000, USA, 2004. Society for Industrial and Applied Mathematics.

16 A. E. F. Clementi, A. Monti, and R. Silvestri. Distributed broadcast in radio networks of unknown topology. *Theoretical Computer Science*, 302:337–364, 2003.

17 G. De Marco and D. Kowalski. Fast nonadaptive deterministic algorithm for conflict resolution in a dynamic multiple-access channel. *SIAM J. Comput*, 44(3):868–888, 2015.

18 Robert G. Gallager. A perspective on multiaccess channels. *IEEE Trans. Information Theory*, 31(2):124–142, 1985.

19 A. G. Greenberg, P. Flajolet, and R. E. Ladner. Estimating the multiplicities of conflicts to speed their resolution in multiple access channels. *Journal of the ACM*, 34(2):289–325, 1987.

20 A. G. Greenberg and R. E. Ladner. Estimating the multiplicities of conflicts in multiple access. In IEEE, editor, *Proc. of the 24th Annual Symp. on Foundations of Computer Science (FOCS) (Tucson, AZ.)*., pages 383–392, Tucson, AZ, USA, 1983. IEEE.

21 A. G. Greenberg and A S. Winograd. lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. *Journal of ACM*, 32:589–596, 1985.

22 J. F. Hayes. An adaptive technique for local distribution. *IEEE Transactions on Communications*, 26:1178–1186, 1978.

23 P. Indyk. Explicit constructions of selectors and related combinatorial structures. In *Proceedings, 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 697–704, San Francisco, CA, USA, 2002. ACM-SIAM.

24 T. Jurdzinski and G. Stachowiak. Probabilistic algorithms for the wakeup problem in single-hop radio networks. *Theory Comput. Syst*, 38(3):347–367, 2005.

25 J. Komlós and A. G. Greenberg. An asymptotically optimal nonadaptive algorithm for conflict resolution in multiple-access channels. *IEEE Trans. on Information Theory*, 31:302–306, 1985.

26 D. Kowalski. On selection problem in radio networks. In *Proceedings, 24th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 158–166, Las Vegas, NV, USA, 2005. ACM.

27 Dariusz R. Kowalski and Andrzej Pelc. Time of deterministic broadcasting in radio networks with local knowledge. *SIAM Journal on Computing*, 33(4):870–891, 2004. doi:10.1137/S0097539702419339.

28 G. De Marco and G. Stachowiak. Asynchronous shared channel. In Elad Michael Schiller and Alexander A. Schwarzmann, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 391–400, Washington, DC, USA, 2017. ACM. doi:10.1145/3087801.3087831.

29 Gianluca De Marco. Distributed broadcast in unknown radio networks. *SIAM J. Comput.*, 39(6):2162–2175, 2010. doi:10.1137/080733826.

30 Gianluca De Marco, Tomasz Jurdzinski, and Dariusz R. Kowalski. Optimal channel utilization with limited feedback. In Leszek Antoni Gasieniec, Jesper Jansson, and Christos Levcopoulos, editors, *Fundamentals of Computation Theory - 22nd International Symposium, FCT 2019, Copenhagen, Denmark, August 12-14, 2019, Proceedings*, volume 11651 of *Lecture Notes in Computer Science*, pages 140–152. Springer, 2019. doi:10.1007/978-3-030-25027-0_10.

31 Gianluca De Marco, Tomasz Jurdzinski, Dariusz R. Kowalski, Michal Rózanski, and Grzegorz Stachowiak. Subquadratic non-adaptive threshold group testing. *J. Comput. Syst. Sci.*, 111:42–56, 2020. doi:10.1016/j.jcss.2020.02.002.

32 Gianluca De Marco, Tomasz Jurdzinski, Michal Rózanski, and Grzegorz Stachowiak. Subquadratic non-adaptive threshold group testing. In Ralf Klasing and Marc Zeitoun, editors, *Fundamentals of Computation Theory - 21st International Symposium, FCT 2017, Bordeaux, France, September 11-13, 2017, Proceedings*, volume 10472 of *Lecture Notes in Computer Science*, pages 177–189. Springer, 2017. doi:10.1007/978-3-662-55751-8_15.

33 Gianluca De Marco and Dariusz R. Kowalski. Towards power-sensitive communication on a multiple-access channel. In *2010 International Conference on Distributed Computing Systems, ICDCS 2010, Genova, Italy, June 21-25, 2010*, pages 728–735. IEEE Computer Society, 2010. doi:10.1109/ICDCS.2010.50.

34 Gianluca De Marco and Dariusz R. Kowalski. Contention resolution in a non-synchronized multiple access channel. In *27th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2013, Cambridge, MA, USA, May 20-24, 2013*, pages 525–533. IEEE Computer Society, 2013. doi:10.1109/IPDPS.2013.68.

35 Gianluca De Marco and Dariusz R. Kowalski. Contention resolution in a non-synchronized multiple access channel. *Theor. Comput. Sci.*, 689:1–13, 2017. doi:10.1016/j.tcs.2017.05.014.

36 Gianluca De Marco, Dariusz R. Kowalski, and Grzegorz Stachowiak. Brief announcement: Deterministic contention resolution on a shared channel. In Ulrich Schmid and Josef Widder, editors, *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, volume 121 of *LIPIcs*, pages 44:1–44:3. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.DISC.2018.44.

37 Gianluca De Marco, Dariusz R. Kowalski, and Grzegorz Stachowiak. Deterministic contention resolution without collision detection: Throughput vs energy. In *41st IEEE International Conference on Distributed Computing Systems, ICDCS 2021, Washington DC, USA, July 7-10, 2021*, pages 1009–1019. IEEE, 2021. doi:10.1109/ICDCS51616.2021.00100.

38 Gianluca De Marco, Marco Pellegrini, and Giovanni Sburlati. Faster deterministic wakeup in multiple access channels. *Discret. Appl. Math.*, 155(8):898–903, 2007. doi:10.1016/j.dam.2006.08.009.

39 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. doi:10.1017/CBO9780511814075.

40 B. S. Tsybakov and V. A. Mikhailov. Free synchronous packet access in a broadcast channel with feedback. *Prob. Inf. Transmission*, 14:259–280, 1977.

A Appendix

A.1 Lemma 1

Proof of Lemma 1. Let us consider the first $32qk$ rounds, for some constant $q > 0$, following the round at which the first station wakes up and starts the computation. We will prove that by the end of this interval the wake up has been accomplished whp(k).

Following the algorithm, each awake station, starting from the round at which it wakes up, transmits using the following sequence of probabilities: $q \cdot \frac{1}{2q}$, $q \cdot \frac{1}{2q+1}$, $q \cdot \frac{1}{2q+2}, \ldots$ If we denote by p_i the transmission probability of an arbitrary awake station u at the ith round of its computation, we have that the sum of transmission probabilities of u over the first $32qk$ rounds, after waking up, is

$$s(32qk) = \sum_{i=1}^{32qk} p_i \leq q \left(\sum_{i=1}^{32qk} \frac{1}{i} \right) \leq q(1 + \ln(32qk)) \ , \tag{2}$$

where in the last step we have used the right-hand inequality of the following known bounds for the hth partial sum H_h of the harmonic series:

$$\ln(1+h) \leq H_h = \sum_{i=1}^{h} \frac{1}{i} \leq 1 + \ln h \ . \tag{3}$$

For any fixed round t, let us consider now the sum of transmission probabilities of all awake stations at time t, denoted as in the previous section by $\sigma(t)$. Since at most k stations can be awake in each round, the average sum $\sigma(t)$ for t ranging over our interval of $32qk$ rounds will be

$$\frac{1}{32qk} \sum_{t=0}^{32qk} \sigma(t) \leq \frac{s(32qk) \cdot k}{32qk} \leq \frac{q(1 + \ln(32qk)) \cdot k}{32qk} \leq \frac{\ln(32qk)}{16} \ .$$

Of course, in at least half of the interval, $\sigma(t)$ must be not larger than twice the average; therefore there is a set T of at least $16qk$ rounds such that

$$\sigma(t) \leq \frac{\ln(32qk)}{8} \ , \tag{4}$$

for every $t \in T$. Let us consider only rounds in T. We say that a round t is *heavy* when $\sigma(t) > 1/2$ and *light* otherwise. We distinguish two cases.

Case 1: there are at least $8qk$ heavy rounds. Recalling also (4), in at least $8qk$ rounds t, it holds that

$$\frac{1}{2} < \sigma(t) \leq \frac{\ln(32qk)}{8} \ . \tag{5}$$

In [24] (cf. Corollary 5.3.1) it is showed that if (5) holds, than the success probability at round t is at least

$$\left(\frac{1}{16}\right)^{\frac{\ln(32qk)}{8}} \geq \frac{1}{\sqrt{32qk}}\ .$$

Therefore, the probability that the wake-up does not appear in $8qk$ heavy rounds is at most $(1-1/\sqrt{32qk})^{8qk} = O(1/k^a)$ for an arbitrary constant a depending on q.

Case 2: there are less than $8qk$ heavy rounds. So, there are at least $\delta = 8qk$ light rounds in T. Let $t_1, t_2, \ldots, t_\delta$ be the sequence of consecutive light rounds. In [24] (cf. Claim 1 in the proof of Theorem 10.3) it is showed by induction on i that $\sigma(t_i) \geq q/(2q+i)$, for $1 \leq i \leq \delta$. Consequently,

$$q/(2q+i) \leq \sigma(t_i) \leq 1/2, \text{ for } 1 \leq i \leq \delta.$$

By Corollary 5.3.3 in [24], this implies that the probability of successfully waking up in the ith light round is at least $q/2(2q+i)$. Hence, the probability that the wakeup is not successful is at most

$$\prod_{i=1}^{\delta}(1-\sigma(t_i)) \leq \prod_{i=1}^{\delta}\left(1-\frac{q}{2(2q+i)}\right) \leq \left(\frac{1}{e}\right)^{\sum_{i=1}^{\delta}\frac{q}{2(2q+i)}} \leq \left(\frac{1}{e}\right)^{\frac{q}{2}\left(\sum_{i=1}^{\delta}\frac{1}{i}-\sum_{i=1}^{2q}\frac{1}{i}\right)}$$
$$= \left(\frac{1}{e}\right)^{\frac{q}{2}(H_\delta - H_{2q})} \leq \left(\frac{1}{e}\right)^{\frac{q}{2}(\ln(1+\delta)-\ln(4q))} = \left(\frac{4q}{1+8qk}\right)^{q/2} \leq \left(\frac{1}{2k}\right)^{q/2},$$

where the second last inequality follows by (3) and choosing $q \geq e/2$).

If in the above arguments the contention k is kept but the number of considered rounds is extended by a factor $\log(k'/k)$, the probability of failure in the formulas is raised to $\log(k'/k)$, which results in success whp(k').

We can observe that Protocol 3 gives a maximum number of transmissions per station of $O(\log k)$ whp. This follows from the fact that in expectation the sum of $q/(2q+i)$ over i up to $O(k)$ is $O(\log k)$. The Chernoff bound and then the union bound over the stations give the desired $O(\log k)$ bound whp(k). The same argument gives energy $O(\log k')$ bound whp(k') when the length of the execution is $O(k\log(k'/k))$. ◀

A.2 Lemma 5

In order to prove Lemma 5 we need to show concentration bounds on the lengths of the activity intervals. To this aim, we will use, similarly as in [2], the Azuma-Hoeffding inequality on martingales.

▶ **Definition 15.** *A sequence of random variables $X_0, X_1, \ldots$ is said to be a martingale sequence if for all $i > 0$, $E[X_i|\ X_0, \ldots, X_{i-1}] = X_{i-1}$.*

The following theorem holds (see e.g. [39, p. 92]).

▶ **Theorem 16** (Azuma-Hoeffding inequality)**.** *Let $X_0, X_1, \ldots$ be a martingale sequence such that for each i,*

$$|X_i - X_{i-1}| \leq c_i,$$

where c_i may depend on i. Then, for all $t \geq 0$ and any $\lambda > 0$,

$$\Pr(|X_t - X_0| \geq \lambda) \leq 2\exp\left(-\frac{\lambda^2}{2\sum_{i=1}^{t} c_i^2}\right).$$

Proof of Lemma 5. Fix a sufficiently large positive integer α. Pick any integer j, where $1 \leq j \leq y$, and consider three cases covering all possible events (i.e., for each j, at least one of the three cases holds):

Case (a): 2^j is at least $\bar{\kappa}^{1/\alpha}$. In such a case, 2^j is a polynomial in $\bar{\kappa}$. Therefore, each ℓ_i for $i \in L_j$ is, by the definition of L_j, upper bounded by $c \cdot 2^j$ whp$(\bar{\kappa}^{1/\alpha})$. Since α is a constant, we can set a suitable parameter $\eta > 0$ in the definition of whp such that the upper bound $\ell_i \leq c \cdot 2^j$ also holds whp$(\bar{\kappa})$. There are at most $\bar{\kappa}$ of such events, by the definition of $\bar{\kappa}$. By applying the union bound to these events, and still choosing a suitable parameter $\eta > 0$ in the definition of whp, we get that $\sum_{i \in L_j} \ell_i$ is $O(2^j w_j)$ holds whp$(\bar{\kappa})$.

Case (b): case (a) does not hold and w_j is at least $\bar{\kappa}^{3/\alpha}$. Let $\mathcal{P}$ be the conditional probability space restricted to all situations where the following event $\mathcal{E}$ holds: for all $i \in L_j$, $\ell_i \leq c \cdot 2^j \log(\bar{\kappa}/2^j)$. Let $i_1, i_2, \ldots, i_{w_j}$ be any order of the indices of L_j. We can define a sequence of random variables $X_{i_1}, X_{i_2}, \ldots, X_{i_{w_j}}$ such that for $1 \leq v \leq w_j$, $X_{i_v} = \ell_{i_1} + \cdots + \ell_{i_v} - vc \cdot 2^j \log(\bar{\kappa}/2^j)$.

We can now observe that in the conditional probability space $\mathcal{P}$ the above sequence of random variables is a martingale. Indeed, we have $E[X_i |\, X_0, \ldots, X_{i-1}] = X_{i-1} + E[X_i] = X_{i-1} + E[\ell_{i_1}] + \cdots + E[\ell_{i_v}] - vc \cdot 2^j \log(\bar{\kappa}/2^j) = X_{i-1}$. We have also that $|X_i - X_{i-1}| = |\ell_{i_v} - c \cdot 2^j \log(\bar{\kappa}/2^j)| = O(\bar{\kappa}^{1/\alpha} \log(\bar{\kappa}))$, where the last step follows because in $\mathcal{P}$ we have that $\ell_i \leq c \cdot 2^j \log(\bar{\kappa}/2^j)$ holds for all $i \in L_j$ and we are assuming that case (a) does not hold, so $2^j < \bar{\kappa}^{1/\alpha}$. Thus the assumptions of Theorem 16 are satisfied and we can estimate the probability that $\ell_{i_1} + \cdots + \ell_{i_{w_j}}$ is larger than $E[\sum_{i \in L_j} \ell_i] + w_j = O(2^j w_j) + w_j = O(2^j w_j)$. Specifically, the Azuma-Hoeffding inequality in our case implies that this probability, within the conditional probability space $\mathcal{P}$, is at most

$$2 \exp\left(-\frac{w_j^2}{2\sum_{i=1}^{w_j} O(1)}\right) = 2 \exp\left(-\frac{w_j^2}{2 w_j O(\bar{\kappa}^{1/\alpha} \log(\bar{\kappa}/2^j))}\right).$$

Since w_j is at least $\bar{\kappa}^{3/\alpha}$, the complementary event holds whp$(\bar{\kappa})$ in the conditional probability space $\mathcal{P}$.

Recall that $\mathcal{E}$, defining the conditional probability space $\mathcal{P}$, is the event that for all $i \in L_j$, $\ell_i \leq c \cdot 2^j \log(\bar{\kappa}/2^j)$. By the definition of L_j, $\ell_i \leq c \cdot 2^j \log(\bar{\kappa}/2^j)$ whp $(\bar{\kappa})$. Hence, the probability that event $\mathcal{E}$ does not hold (i.e., the case is outside $\mathcal{P}$) is, by the union bound, at most $O(1/\bar{\kappa})$ for $\alpha > 3$. This implies that $\sum_{i \in L_j} \ell_i \leq O(2^j w_j)$ holds whp$(\bar{\kappa})$ in the whole sample space.

Case (c): cases (a) and (b) do not hold, i.e., $2^j < \bar{\kappa}^{1/\alpha}$ and $w_j < \bar{\kappa}^{3/\alpha}$, which implies $2^j w_j$ is smaller than $\bar{\kappa}^{4/\alpha}$.

Putting all cases together, the total contribution of j satisfying Cases (a), (b) and (c) is linear in $O(\sum_j 2^j w_j) = O(\bar{\kappa})$ and holds whp$(\bar{\kappa})$, by the union bound taken over at most $x \leq \bar{\kappa}$ values j and using upper bound $O(2^j w_j)$ that holds whp$(\bar{\kappa})$ each. ◀